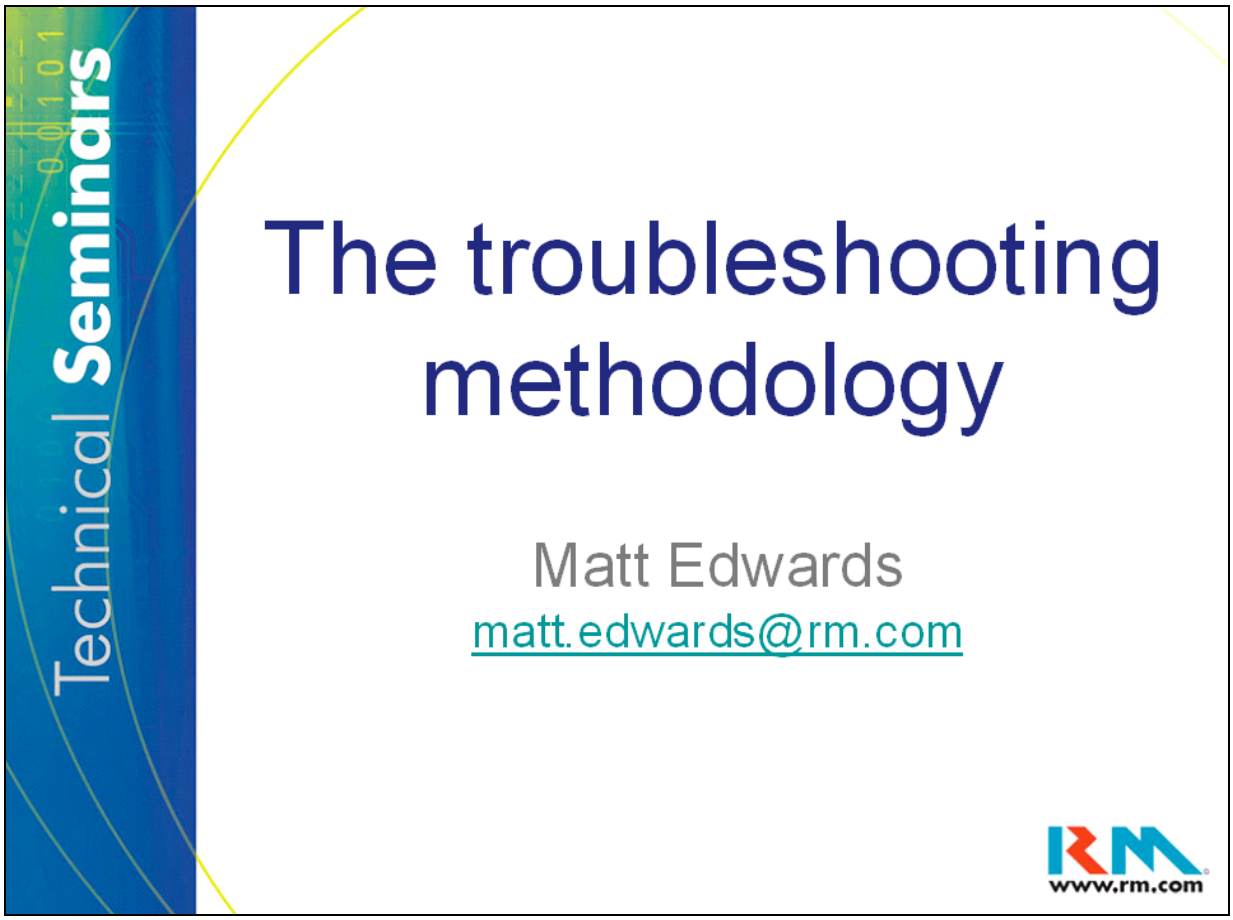
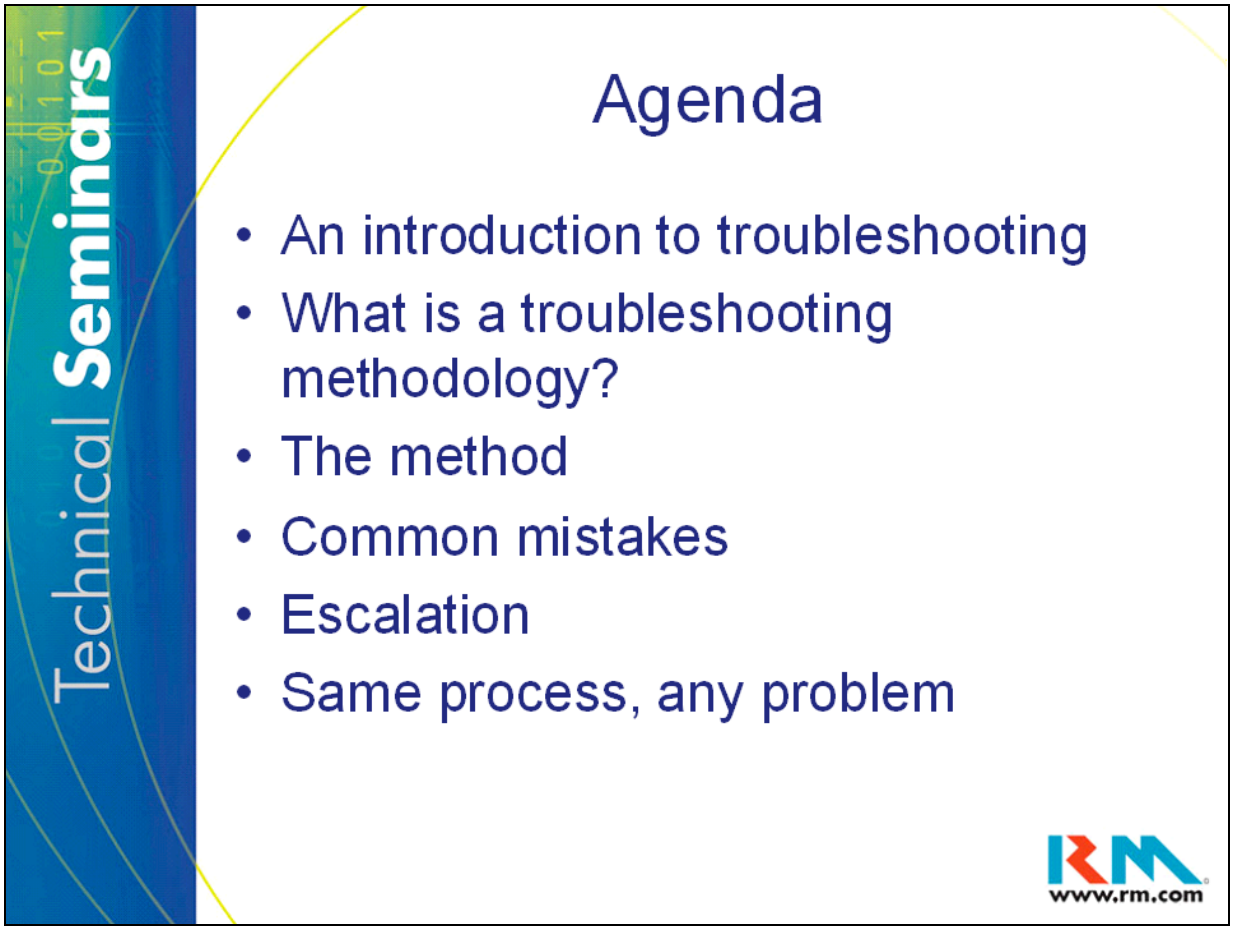


Same process, any problem

© RM 2010


Technical Seminars
Agenda
• An introduction to troubleshooting
• What is a troubleshooting methodology?
• The method
• Common mistakes
• Escalation
• Same process, any problem
RM
www.rm.com

© RM 2010

An Introduction to troubleshooting

Troubleshooting is a skill used by almost everyone in many different parts of our working and personal lives. To be able to troubleshoot an issue quickly and effectively, you need to have a theory of how to approach the issue. You should be able to apply this process to any problem or fault you encounter.

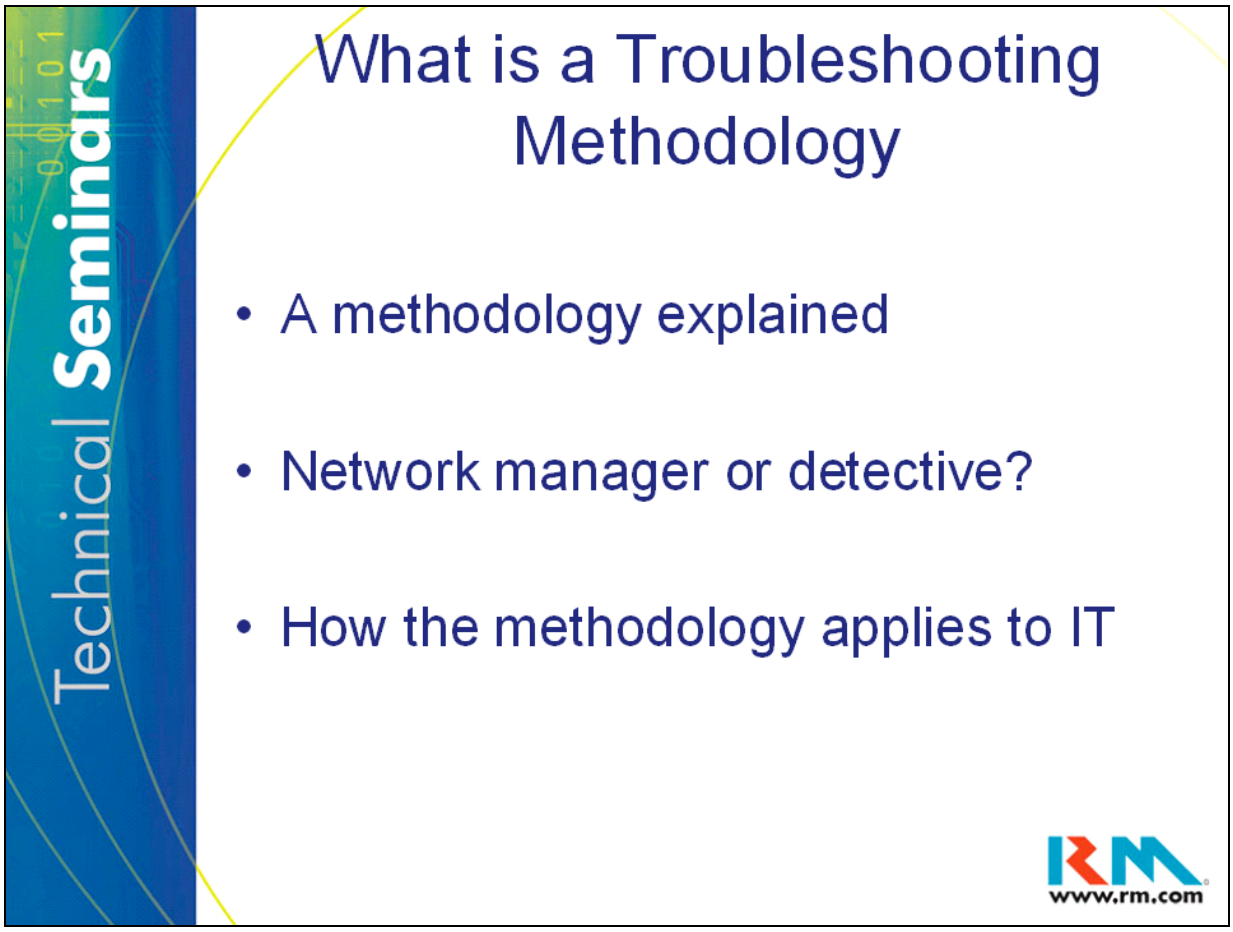
History

There are a few theories on where the term troubleshooting comes from. The Oxford English Dictionary reports that it dates back to the late 19th century and originally related to the repair of telephone lines.
The term was originally 'trouble-hunter' and was given to engineers who would find faults on telephone lines and repair them.

Another theory was that trouble-shooters were first seen in the days of the gold rush (late 19th century) and new settlers. They were private agents who were hired out to help settlers and gold miners protect their rightful and hard-earned property and their families. They earned room and board and in some cases a small reward for being there to do the "shooting" if and when it became necessary. Hence: trouble-shooter.

It hasn't changed that much since then, troubleshooting is really a way finding the cause of a problem and fixing it.

© RM 2010



What is a troubleshooting methodology

I worked as a hardware quality engineer for some years and inspecting faulty hardware coming back from the field was a part of my job. It always surprised me how high the 'no fault found' rate was, or how many machines were diagnosed incorrectly. This got me interested in the theory of how to diagnose faults or problems. It became clear that a lot of people do not think logically about problem solving or troubleshooting an issue. In this session we are going to look at how to apply theory to troubleshooting and the benefits it can bring.

If you've ever been in a job interview and they have given you an example problem and asked for 'the steps you would take to fix the problem', in most cases what they are asking for is your troubleshooting methodology.
A troubleshooting methodology is not the specific actions you would take such as reboot the computer or reinstall a driver; it is an overall strategy for how to approach a problem. In our interview example, the steps you would take.
We are going to look at how to develop a consistent approach for any problem you encounter; using some method to the way you approach a problem or fault will help you get to the root cause much faster than randomly attempting fixes.

You can look at it as detective work and to take this analogy further, think about how the police would solve a crime.

 © RM 2010

- The crime takes place and is reported to the police.
- The police start by gathering information. They will find out who has information about the crime and will interview them.
- A review of that information will then take place.
- A list of possible suspects will be drawn up.
- Starting with the most obvious suspect, they will start ruling out the suspects using evidence and proof. They cannot assume certain facts; everything must be proven before they can rule out a suspect. They look for suspects who could have not committed the crime at first, this quickly reduces the list of suspects to just the ones that 'could' have committed the crime. They now have a shortlist.
- Then using much more in-depth investigations they will analyze each suspect in detail in the most logical order to prove or disprove their involvement.
- These in-depth investigations will normally reveal who committed the crime.
- If not they will go back and review the information to see if anything was missed, taking into account all they had ruled out already. A new list of suspects will be drawn up and the process will be repeated as above.

This is a troubleshooting methodology, as you see I haven't spoken about the specifics of the case in question but we have developed a list of steps to quickly get to the cause.

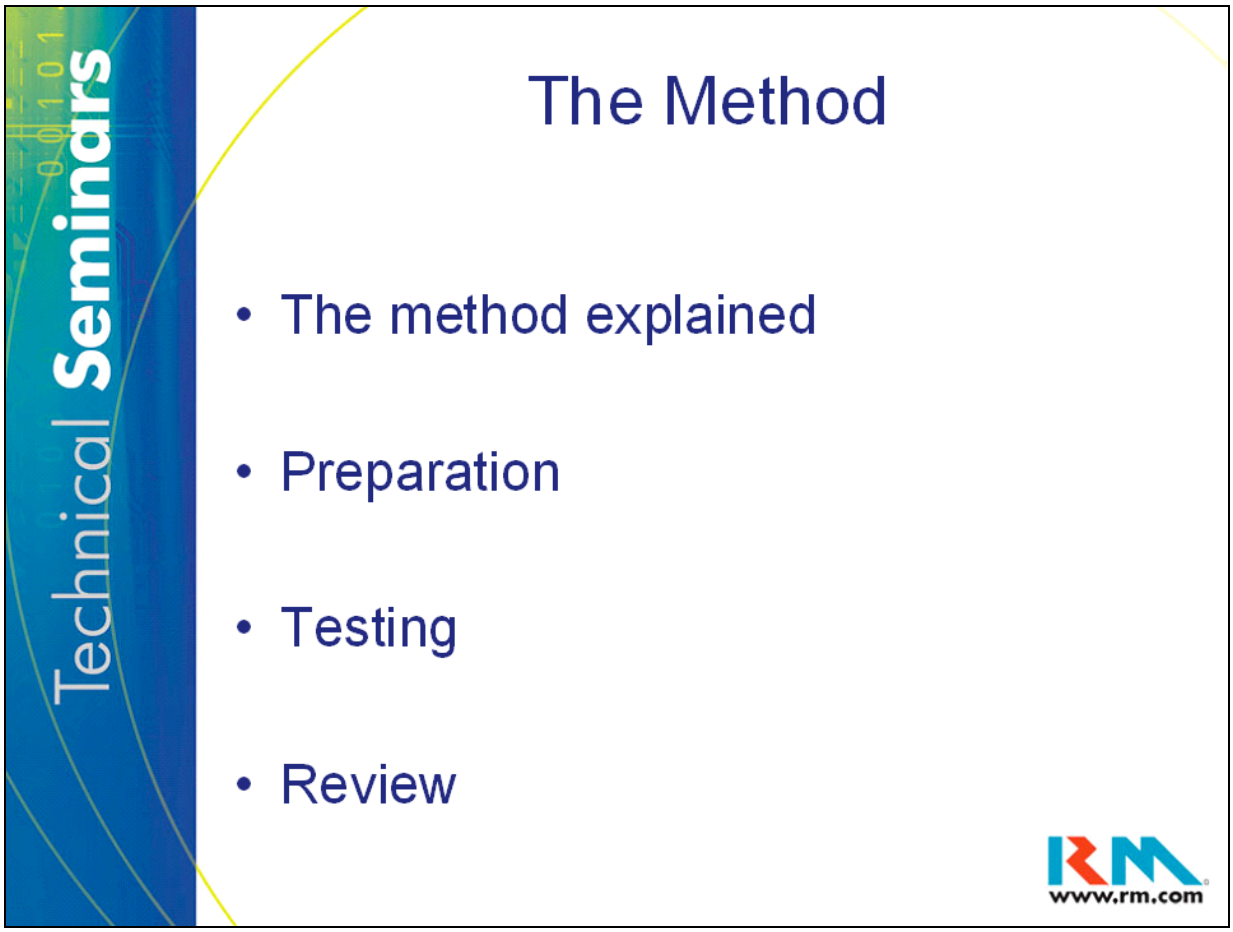
If we look at the above list of steps again and think of it in terms of a computer fault, they apply almost perfectly.

- The fault takes place and is reported to the technicians.
- The technicians start by gathering information. They will find out who has information about the fault and will interview them.
- A review of that information will then take place.
- A list of possible causes will be drawn up.
- Starting with the most obvious cause, they will start ruling out the suspects (software/hardware) using evidence and proof. They cannot assume certain facts; everything must be proven before they can rule out a suspect. They look for suspects that could have not caused the fault at first, this quickly reduces the list of suspects to just the ones that 'could' have caused the fault. They now have a shortlist.
- Then using much more in-depth investigations they will analyze each suspect in detail in the most logical order to prove or disprove their involvement.
- These in-depth investigations will normally reveal what caused the fault.
- If not they will go back the review of the information to see if anything was missed, taking into account all they had ruled out already. A new list of suspects will be drawn up and the process will be repeated as above.

You can apply the same logic to diagnosing a medical issue or a mechanical engineering fault. This is an example of how to apply theory to a problem. Without applying theory to your fault finding you are looking for the needle in the haystack.
Next we are going to look at what the general steps you should be taking are and what advantages they bring.

 © RM 2010

© RM 2010



The method

A lot of the fault finding or troubleshooting done by network technicians/managers uses skills that have been built up by experience of dealing with many different faults. You cannot teach anyone all the faults that they may encounter when supporting a network, but you can teach them the method of how to attack an issue, whatever it is.
By having a set methodology in mind when approaching a fault, you are giving yourself the assurance of not missing anything. Even the most experienced engineers will have come across faults where they miss something early in the diagnosis due to a hunch or gut feeling about what is causing the fault. Any technician or engineer that claims they have never missed an obvious cause to a fault is either lying or is a cyborg.
I can remember many times when trying to find the root cause of a fault, going off down the wrong road due to me assuming at the start I knew what was causing the fault and days later realising I had missed something extremely obvious. I am not going to give you specific examples as these still bring me some embarrassment all these years later!

I was introduced to the idea of learning a set of skills to minimise the chance of making these mistakes.
Now at this point you might be worried that I am going to start using words like *flowchart* or *question set*, I can promise you that is not what I am talking about. Flowcharts and question sets have their place in the diagnostics family tree and are an extremely useful for first line support technicians but for fixing more complex issues it helps to have the skills required in your head.

© RM 2010

So let's have a look at our troubleshooting methodology in detail, looking at the order in which you should attack a problem and the reasons why these steps are so important.

Gather information/the interview

This really should be your first point of call. A fault will generally be reported to you via an email or a telephone call, at this point some information will be passed across. It is up to you to decide when you have enough information about a fault before you move on to trying to fix it, the simplest way to do this is to conduct an interview with anyone who had an impact or any involvement with the fault in question.
So when I say interview I don't mean you have to take the person who was using the computer off to a meeting room and sit them down under bright lights to get what you need out of them - well you could try that!
The purpose of the interview is for you to gain a list of specific information relating to the fault from the user before you can properly investigate the fault. In some cases you will need to explain this to the user as they may feel that you are not trying to fix the problem straight away.
I would strongly advise writing down the answers to your questions, collating this information about the fault could benefit you hugely later on if you need to go back and review what has been done, or review how to reproduce the fault. Another benefit is; if you are working in a team of technicians it is good working practise for your team to be able to review details of any fault found so knowledge can be shared within the team.

What is it we need from the users then? Below is a list of questions you should think about when a fault is reported, this is what I would encourage you to be asking users. Some of the answers to these will be made clear when the fault is reported so ask them at your discretion, just be sure to make a note if something is reported in the fault description.
These generic questions not only collate very useful pieces of data about where a fault might reside, but in some cases they will actually lead you right to the front door of where the fault lives.

Eliminate possible PEBKAC errors
So what is a PEBKAC error? Well it is a slightly unflattering term which stands for '**P**roblem **E**xists **B**etween **K**eyboard **A**nd **C**hair'. It means the fault is caused by the user's actions rather than a hardware or software fault. These questions should help unearth any PEBKAC errors. You will see a lot of these working in IT; you might even cause the odd one!

What were you trying to do?
What was the user trying to run or access? They might imply the computer is faulty but there may be a network restriction stopping them accessing what they want. Ask them the exact steps they took to try and achieve what they were doing. Make sure they are taking the correct steps. Think PEBKAC.

 © RM 2010

Did it ever work?
Very useful to know this, for example; did the software ever run?, was audio working out of the headphones previously? You are trying to find out if something was working and has then failed or if something has never worked, if so there is a good chance it has been configured incorrectly, you should check this first.

What exactly were you doing at the time of the fault?
You need this to be as specific as possible, try and ask the user to trace their steps. If it is an error code that is appearing when trying to access software, take down the exact error letter for letter. If possible talk them through how to take a screenshot and send it through. You may see something in the screenshot which may point towards why this software is not running and you also get to see the error accurately.

Date and time
Fairly self explanatory but do make sure you record dates and times of when faults occur. Some faults will only occur in a certain time window due to their cause, you may miss out on knowing this if you have not recorded the date and time.

Can you try it on another computer?
Ask the user to try and access what they are trying to get to from another computer in the room if possible. In most cases by doing this you are straight away finding out if the fault is local to the computer or if it's a more widespread network issue. Users sometimes get grumpy at having to go and try this but explain the benefit of this step; it really does narrow down your list of suspects.

Has anything changed?
Your interview doesn't just relate to the user of the computer but anyone that may have had an impact. You need to speak to your other network technicians to see if anything new has been installed on that computer recently or if it has been rebuilt. There may be a new hardware component or a new application installed that you don't know about which may relate to the fault in question.

There are other things you may want to consider when collating information but they would be more specific steps that may relate to your environment. Things such as location, is it a mobile computer, serial numbers, username.
The steps above are all things which you should consider asking anyone that you feel could have had an impact on the fault. Yes, I know I keep going on about it but please.....write down as much of this information as possible.

Information review

Once you have collated and written down all of the information relating to when the fault happens and if the environment has changed in anyway, you then need a review of this information.
One piece of advice I would give you at this stage, is to get the fault into a bullet point format (example below) and then tell someone else about the fault. This may sound a little odd but it is the way the human brain works, it helps you to fully understand a problem and how to deal with it if you have to relay or explain it to someone else face to

 © RM 2010

face. Once you are explaining it to them you may well start to realise the obvious next course of action, also by doing this you will get a second opinion on the fault which should never be discounted. I can vouch for this technique and with any non obvious fault I always find someone to go and tell about the issue, not because I think they can fix it for me but I find it really helps me work out where to go next by explaining it to someone else.

Example

- I've got an issue where the network card connection drops off at random times on one computer.
- The machine has been working fine for over a year.
- Can drop connection at any time, sometimes the user can't log on in the morning after first boot up.
- All other machines in the room seem fine.
- No new software or hardware has been installed recently.

By having a conversation about the fault with someone else can really help you get it clear in your mind the next best course of action.
At the end of this review of information you should have a rough list of possible causes and you should have a list of things you can already rule out. Now we get into how to troubleshoot the list of possible causes.

Google it

It sounds ridiculous but I am including this as a side point. You would not believe the amount of computers Google has fixed for me (other internet search engines are available blah blah blah). I'm not ashamed, honestly!

Often I have been sent computers which are displaying an error either during boot or within Microsoft® Windows®, I go to Google and type in the error message exactly word for word. Not always, but certainly with windows application errors you will often find information relating to the fault or walkthroughs on how to resolve it. It always surprised me that the users weren't doing this themselves, as in a lot of cases the fix may be a driver update or hotfix which is easily downloaded and installed.
Now I must say that although it can be useful for gaining information about a fault, some information on the Internet can be dangerously wrong. Be wary of information in random forums, it's often unfounded opinion rather than tested theory.
To go back to my Police analogy, you should see Google (and other such search engines, yawn) like a police informant. They will give you strong opinion which stands a good chance of being true, but it is up to you to consider that information, create your own hypothesis and prove the theory through testing and evidence.

Basically put, you should consider Internet search engines to give you clues at the start of a diagnosis or when you hit a wall and are not sure what to try next. Always create your

 © RM 2010

own list of suspects alongside Internet research as blindly following a resolution from an Internet forum can sometimes get you in more trouble than when you started.

Testing

Once you have a list of possible suspects you will need to start ruling some of them out. To do this, you reverse what you would naturally do; don't look for what is causing the fault, look for what is not causing the fault.
This is a technique used a lot in hardware diagnostics. At the early stages of diagnosing a fault this is the quickest way of turning your rough list of possible suspects into a shortlist. If you can prove what is not causing the fault, your shortlist will make itself.

There are some rules you should use when fault finding, a set of guidelines you should always stick to.

I have known engineers who do not apply these rules and I can promise you they go the long way round to get the root of a fault. They prefer to rely on a hunch or using knowledge from a fault they may have seen before, now of course this is fine if it really is an issue you have dealt with many times. I am not saying don't use any intuition or experience but it has always surprised me how many engineers use their hunch when fault finding unknown faults or more commonly, faults with many possible causes.

These are the rules I would use when fault finding any issue, software or hardware:

- Be able to consistently recreate the symptom/fault.
- Only change one thing at a time.
- Test it (recreate the symptom).
- Write down the results.
- If the symptom still exists, change the environment back to it's original failing state.

Now let's have a look at these in more detail.

Be able to consistently recreate the symptom/fault

This is without doubt the first thing you should do when looking at a fault or any issue. You need to know what steps it takes to induce the fault. You should be able to get a lot of this information from the user who originally experienced the symptom. Ask them the steps they took right before they saw the symptom, then try and recreate these steps to see if you can consistently recreate the fault. If you cannot recreate the symptom on demand you are going to struggle to know if any fixes you apply have made any difference. It can sometimes be time consuming but I would advise putting in as much time as required to consistently recreating the fault. Once you can recreate the symptom on demand, write these steps down in case you need them in future.

 © RM 2010

Only change one thing at a time

An easy mistake to make this one. A big part of troubleshooting is finding out what was causing the issue, not just making the problem go away. Once you can recreate the fault consistently you can start to run tests and diagnostics to try and fix it. If you want to know what was causing the issue it is extremely important to only change one thing at a time. If you change two or three parts of the environment at the same time and then try and recreate the symptom, you may find you have fixed it.....great. Well that is until the problem arises further down the line, you will not actually know what caused the issue. Fixing the issue is one achievement, knowing what caused it and putting in steps to make sure it doesn't happen again is the main goal of troubleshooting.

Test it

Basically put, go back to the start. Once you have changed one part of the environment (as above), you need to go back and recreate your symptom again. If the fault still exists you know the change you made does not affect the symptom and you can move onto the next suspect.

Write down the results

I cannot advise you strongly enough to write the results of this testing down. It serves a few purposes. Firstly for you, it's important to remember what you have done; some faults can take a few weeks or months to diagnose and you may want to go back and review the steps you have already taken. Secondly it's important in case you ever need to pass on the testing you've done to someone else. This may be an escalation to someone else in your establishment or it may be an escalation to a supplier or manufacturer. They will need to know what has been tried already so they do not repeat your steps. We are going to look at escalations a little later on in this session.

It the symptom still exists, change the environment back to it's original failing state

So we've changed one thing, we've rerun the test to recreate the symptom and the fault still remains. We've written down the results of what we changed and what happened on the retest of the symptom. It is vital at this stage to revert the change you made back to its original state. The reason for this is that the change you made may not have made any difference to the symptom but it may have an effect on anything else you change from here on in. If possible the computer should be in its original failing state for every test you run, if you are not resetting everything back to its original state then by the time you get four or five tests in, the environment you are testing is not comparable with the original environment the user saw the symptom under. It seems like a lot of work but the more thorough you are about the test environment, the more accurate your results will be.

 © RM 2010

Ok let's have a look at a common symptom and how we would go about troubleshooting it.

You have a fault reported where a computer is now showing no display.
The system had been working fine the day before.
It is always off, they have rebooted it but the screen is off and they never see a logon screen.
You asked them to check that all the connections look plugged in correctly and they confirm that they are. Also you have checked the monitor is powered on correctly.

So when we are happy there is nothing obvious causing the no display. We should now be able to think about a rough list of suspects:

Corrupt software
Faulty monitor
Faulty cabling
Faulty hardware (graphics adaptor/mainboard)

These are listed in the order you should be checking them for this example. There are of course multiple causes under each of these 4 areas and you need to work out how best to start testing these. To follow through with this example let's have a quick look at how we would diagnose the most obvious causes step by step.

Corrupt software – To test this I would reboot the computer and check if you get a display during boot up (POST). Do you see a BIOS splash screen? Can you boot the system up in safe mode? If you can see a display at BIOS level but then lose the display after this, you can be almost positive your hardware is not faulty. This indicates that when the video driver for Windows is loaded, it is corrupt and cannot load a display. In the case of our example, if the driver was corrupt you would discover it at this point when you either, check if there is a visible display at BIOS level or boot to Windows safe mode. If you do not get a display at BIOS level, move onto the next step.

Faulty monitor – I would suggest always testing the components that are easiest to test first. If you are getting no display and you're happy the software image isn't corrupt, the next easiest thing to test would be the monitor. This is known as a swap test. Take a component (in this case monitor) that you know works and swap it with the component (monitor) involved in this computers setup. Keep the cabling the same, remember the importance of only changing one thing at a time. You will know if you have a faulty monitor at this point, if the known working monitor also does not show a display then you have localised the fault to your computers cabling or hardware. Don't forget to change the monitor back; resetting everything back to the original environment as much as possible after every test is always advised.

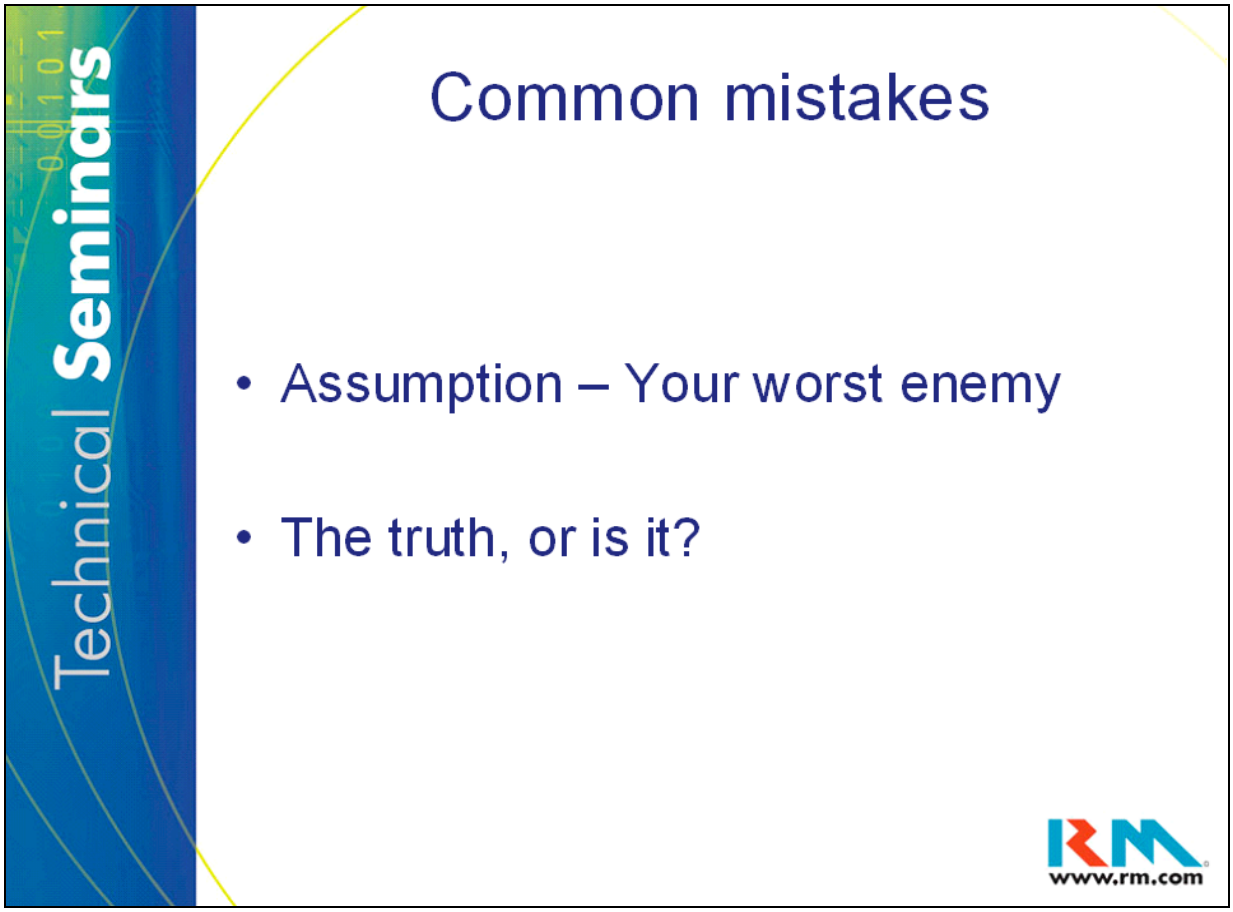
Faulty cabling – Exactly the same process as the above monitor test. Swap test a known working video cable. Again if the new cable does not work then the fault resides

 © RM 2010

somewhere in the computer's hardware, if the new cable works then your fault was a damaged or failing cable.

Faulty hardware – Now you have ruled out all of the above, you can be confident contacting your hardware manufacturer to arrange a warranty/chargeable repair of the computer. This is a good example of why you should write down all of your information/diagnostics and tests run, as it will make your life much easier when reporting hardware faults. Your hardware vendor will be very happy that you have run clear, thorough tests and have collated all your results. It also means you'll probably get your engineer a little quicker!
I'm not going to go through how to diagnose the faulty computer component in this session as that is getting into specifics too much.

Above is a very simple example of a fairly common symptom. Hopefully you can see that by thinking first about all the possible causes and then by working out which is the simplest to test; you cover all bases and are much less likely to miss something. This is a good base for how to format your fault finding.



© RM 2010

Common mistakes

Assumption

Pronunciation: *uh*-**suhmp**-sh*uh* n

- A thing that is accepted as true or as certain to happen, without proof.
- The action of taking on power or responsibility.
- Arrogance or presumption.

Assumption is your worst enemy in the IT industry. I have seen enough crazy faults which initially make no sense at all, to know not to trust computers at all anymore!
It is understandable why it happens though, when a problem or fault is reported to us, it is natural that the brain will start to try and work out what has caused it and you will probably end up with an assumption on what is the root cause. Do not disregard this but I would discourage you from running with your assumption as the main suspect unless you are 100% sure you are right.
The main reason I would suggest leaving your assumption aside and going though the logical fault finding process we looked at in the last module is this:

If you start with your assumption you will find (probably without noticing) you will try to prove your assumption was correct and not to find the fault. This is a very common human behaviour and as I say you may not even notice yourself doing it but it will happen. There is an unintended arrogance when it comes to troubleshooting problems, be it an IT issue, a medical or criminal issue.
Some people just like to prove their initial assumption was correct, in actual fact it doesn't matter who guessed it right at the start, as the point is surely fixing the issue. I can guarantee that if you rely on assumptions and hunches to troubleshoot the issues in your schools; you will get more wrong than you get right. Think tortoise and the hare.

Here are some famously incorrect assumptions made by some rather wise people from history. It shows how you need to be careful what you assume, you could be proved very wrong further down the line:

Charles Duell was Commissioner at the US Patents Office who in 1899 gave his opinion that, "Everything that can be invented has been invented."

General Douglas Haig (1861 -1928) the commander of the British Army in WWI said in 1914 of the machine gun, "Make no mistake, this weapon will change absolutely nothing."

In 1927, H.M Warner of Warner Brothers asked, "Who the hell wants to hear actors talk?"

Rex Lambert, Editor of The Listener magazine, wrote in 1936, "Television won't matter in your lifetime or mine."

 © RM 2010

Don Rowe was the director of Decca Records who turned down the Beatles. He said to their promoter, Brian Epstein, "We don't like your boys' sound. Groups of guitarists are on the way out."

Ken Olson, CEO of DEC (Digital Equipment Corporation) said in 1977, "There is no reason anyone would want a computer in their home."

Bill Gates stated in 1981, "640k ought to be enough for anybody."

Avoiding assumptions is a part of every stage of troubleshooting but possibly most important when thinking about fault finding or running diagnostics tests. A common mistake is assuming a fault has been resolved as you may have changed something and then retested without seeing the symptom. This comes back to being able to consistently recreate the symptoms. So many faults in IT are intermittent or inconsistent, you can only be sure a fault is resolved through proof and evidence. This proof is made up through creating an accurate shortlist of suspects and using the 5 rules of testing we looked at earlier.

- Be able to consistently recreate the symptom/fault.
- Only change one thing at a time.
- Test it (recreate the symptom).
- Write down the results.
- If the symptom still exists, change the environment back to it's original failing state.

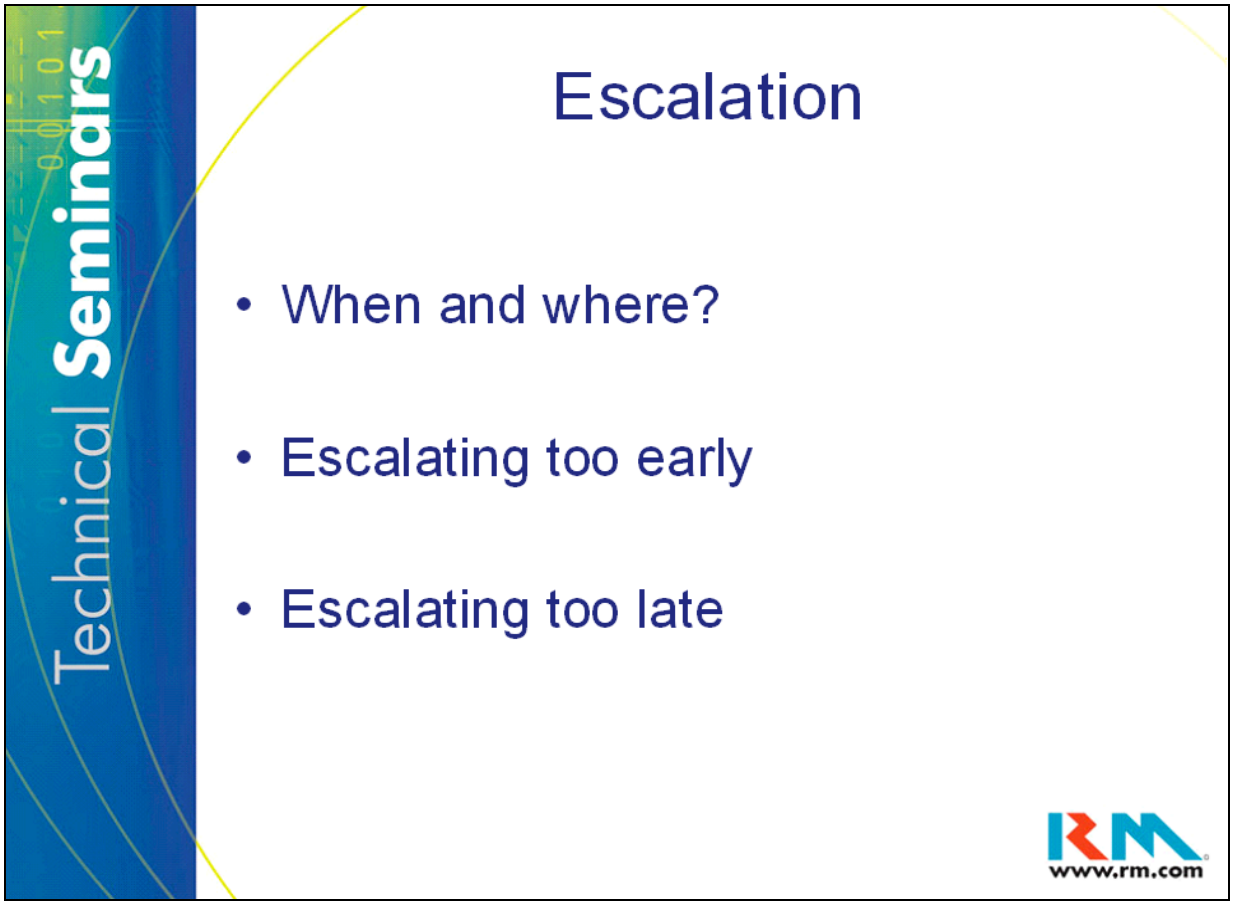
When it comes to troubleshooting computer hardware and software, the best advice I can give you is – Don't assume anything!!!

People lie

Ok, it seems either a bit obvious or a bit harsh but it is true, people lie. This is something you must consider when troubleshooting. The police have to be aware of people lying about their involvement in a crime, or the steps they took. Doctors have to consider people lying about their social habits which may have an effect on a medical symptom. This is no different in the IT world, people do not like to look stupid. You can ask a user "Did you do anything to it?". To be honest you are probably wasting your time even asking this question. You need to think cleverly about how you ask your users your questions, so instead of "Did you do anything to it?" you could try "Can you talk me through where you clicked and what steps you took?" This is much less confrontational and the user will not feel like you are applying blame to them. Users are fragile creatures and they do not like speaking to people who really know what is happening with a computer, they seem much happier living under the opinion that computers have a mind of their own.

 © RM 2010

I guess what I'm saying is, take the information given by users with a pinch of salt. Be aware that they might just be saying what they think you want to hear. If you are unsure, do not confront the user as this will get you nowhere but test the system yourself. If you suspect they have done something to induce an issue or fault, go and test this theory yourself then you can be sure.
A common mistake made by a lot of computer technicians is taking the users word as gospel; I always take the option of confirming what they have told me by repeating their steps myself.



Escalation

In most jobs you will have an escalation point. Someone you can go to with a problem either to discuss, or to pass the problem over for them to take ownership. Escalation is a key step of any support team. If we take RM as an example, let's have a look at how a support call may be handled.

Customer contacts First Line Support and explains the fault.

 © RM 2010

FRG (First Line) may check the Knowledge Library to see if there is an obvious fix detailed in a technical article. If not, they accept this will need to be investigated in more detail and the call will then be escalated to 2nd Line Support.

2nd Line Support will then take on the call and carry out further investigations. They have more time to investigate each call and will attempt to resolve the fault during their investigations. If they are unable to resolve the issue after testing everything they suspected was causing the fault, they would escalate the call the 3rd Line Support.

3rd Line Support will take on the escalation and own the call through to resolution. They may need to contact the development area of the business or contact the manufacturers of components which relate to the fault.

This is a standard escalation process for most IT support teams. There are 2 very common mistakes made when escalating, they are:

- Escalating too early.
- Escalating too late.

Both of these will cause frustration for everyone concerned, so let's have a look at why this is and how we can best avoid escalating issue at the wrong time.

Escalating too early

We have all had problems we wish we could give to someone else. I can certainly think of a few!
You have to put yourself in the shoes of the person who is taking the escalation from you; they are going to expect a certain amount of the work to be carried out already, and rightly so. They will want to see this documented in a clear, easy to read format; I have taken plenty of escalations where all the information was there but it was about 10,000 words long and all in one paragraph! Not helpful!
Escalating too early generally comes from either laziness or lack of clear information relating to the symptom and any testing already carried out. This all comes back to our fault finding rules and our troubleshooting methodology overall. If you have written down all of the stages of your investigations and detailed any testing results you will find the natural point of escalation. This will come when you run out of suspects on your shortlist, when you decide you have tested everything you and your team suspect could be causing the fault then it may be the right time to escalate.
If we come back to our RM Support example above, we use a call logging system so any information passed on from the user at first line level will be documented for any other engineer who may be looking at the call in future. You should apply this same logic to any faults reported in your schools.
You will find it a lot easier to escalate an issue if you can detail every step of your diagnosis so far. Escalating an issue too early basically means you have not done everything you could have done before trying to pass the issue onto someone else.

Escalating too late

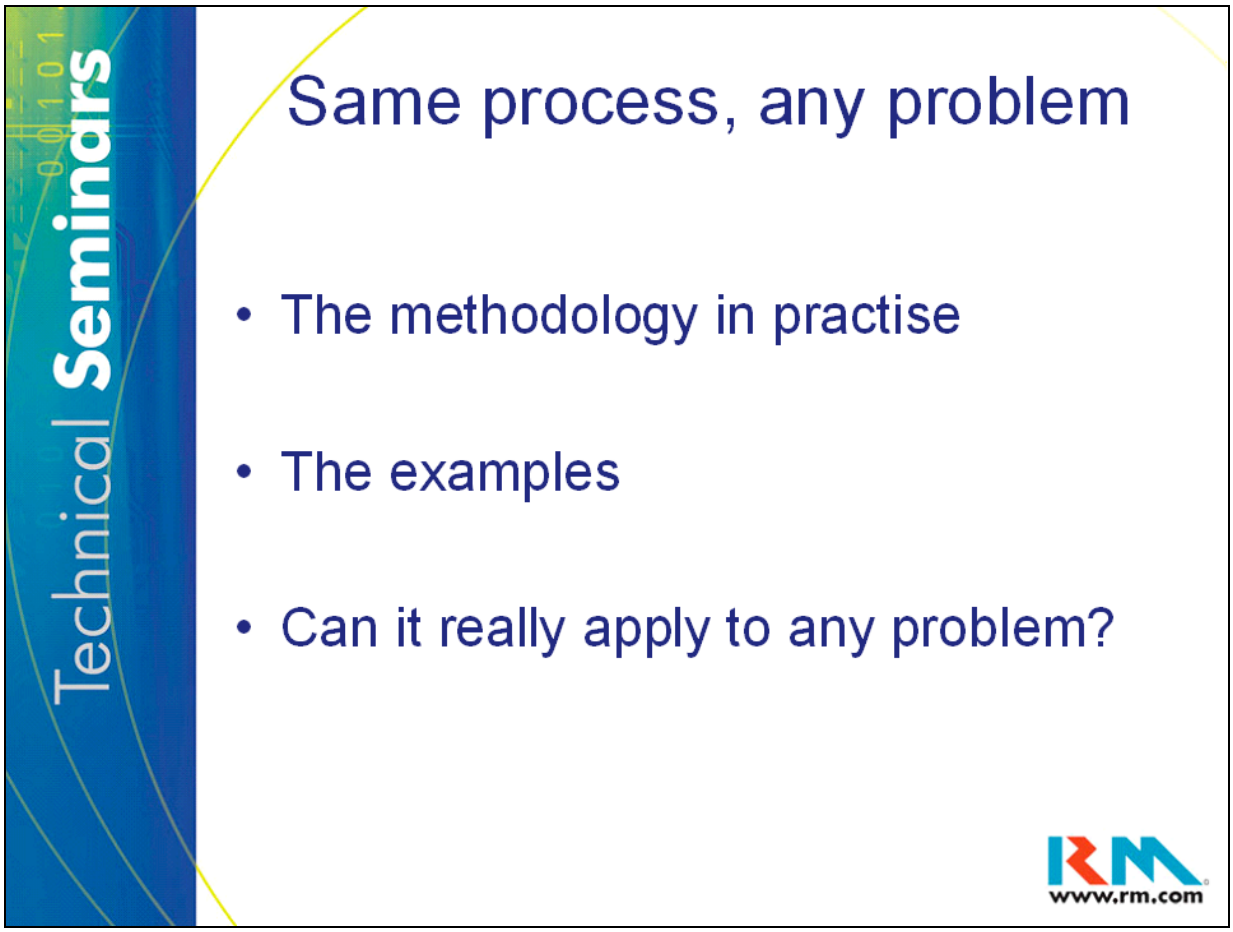
 © RM 2010

Now this is something I suffer from; low level arrogance. Not arrogance in that I think I know everything (although others may tell you different!) but arrogant in that I feel I can always resolve the issue. In short, I mean not knowing when to let go and let someone else take on the issue. It's something you'll see a lot in any environment where fault finding or troubleshooting occurs. Generally people who fix faults or issues in any field are quizzical sorts who are fascinated with why things fail, and they secretly love the challenge of trying to fix them. This can result in holding on to the issue longer than you should; I believe this is down to pride. People like us like to fix things and when we are struggling one of two things will happen. You will either give up, think "I wish this problem would disappear" and maybe escalate to someone else. Others (like me) get angry and stubborn and refuse to be beaten by a computer.

This attitude can lead to problems. I can think of an example where I was investigating a suspected hardware fault but could not narrow down what was causing it. It had a very strange symptom which was hard to recreate. Now, I sat and worked on this in a very dark test lab all on my own for a serious number of weeks. This was something I wanted to get to the bottom of; I wanted to be the hero!
Anyway after a few weeks of not getting much further I was chatting to a colleague at the coffee machine and had mentioned about the difficult issue which was causing me to go mad, he very casually said "oh that sounds like that issue Dave (senior engineer) saw last year."
Well it turned out it was the issue that Dave saw last year and if I had escalated it up to my senior engineers at the point I was struggling, they could have told me what was causing the issue and how to resolve it. It was an extremely complicated fix but it was all documented and had walkthroughs of how to resolve it.
I was young and I learnt the lesson; do not let your personal pride get in the way of getting to bottom of an issue as quickly as possible.

The best engineers I have ever worked with have no ego when it comes to escalating an issue to someone else. They also would keep an interest in the issue even after passing it on so they can find out the resolution should the issue arise again. Just because an issue has been escalated it doesn't mean you should wash your hands of it.

 © RM 2010



Same process, any problem

Ok let's take our troubleshooting methodology and see if we can apply it to a range of problems. Below are 3 examples of issues from different fields. We're going to have a look to see if our troubleshooting methodology can apply. In these examples remember that the method we're talking about is not the specific steps like 'reboot the computer' but the overall strategy for finding the fault.

Example 1
You have a television which is not displaying any channels.

Example 2
You have a home computer which will not connect to the Internet.

Example 3
You are moving a sofa into your new flat and it won't fit through the door.

So where do we start for all 3 examples? As we've looked at previously, before trying to fix a problem we have to first understand what we are trying to achieve and what is happening.

 © RM 2010

Gather information/the interview

What are we trying to achieve for each example?

Example 1 – Watch television.
Example 2 – Browse the Internet.
Example 3 – Get the sofa in the living room!

Scenario	**Gather Information**
TV does not display any channels	Did the TV work previously? What devices are involved in delivering the TV channels? Analogue aerial/Freeview digital box/Cable/Sky Digital/Cabling? Have any settings been changed or anything new been introduced into the environment?
Computer will not connect to the internet	Was the Internet working previously? How does the computer connect to the Internet? Modem/Router/Wifi? What error message is displayed when trying to connect to the Internet?
New sofa will not fit through the front door	What are the sofas height, length and width? What are the doorways dimensions, height and width? What angle is the sofa at when trying to move it through the door?

As you can see we now have some key information about each problem. Even our sofa problem, by measuring the sofa and the door will give some information on whether you are wasting your time and should look at another solution.

We will now have a review of the information and create are shortlists of suspects.

Scenario	**Suspects**
TV does not display channels	Faulty TV Faulty receiver box Signal fault
Computer will not connect to the Internet	Corrupt Windows settings Faulty infrastructure Faulty computer hardware Internet provider fault

 © RM 2010

New sofa will not fit through the front door	You are using the wrong angle The sofa is too large to fit through the door. Is there another entrance to the building, a window/back door?

Now we will start to look into testing these theories. The approach for testing should be the same regardless of what we are testing. Putting in the background work above will help in understanding what it is we are trying to test for.
Remember the 5 rules of testing we looked at earlier:
Recreate the symptom, change one thing at a time, test it, write down the results (if applicable), if the symptom still exists; change it back.

In some cases you may not need to write down results, for example 3 it would not be required but the logical order of testing would still apply.

Scenario	**Test**
TV does not display channels	Test another TV in the house Test another device, DVD/Games console Try another TV through your digital box if possible
Computer will not connect to the Internet	Restart the modem/router Check cabling Check software settings
New sofa will not fit through the front door	Work out and try the best possible angle of approach Measure other entrances to the house, windows/back door Does the sofa break down into 2 pieces?

Although the steps taken are obviously different, you can see that the theory behind them is the same. Test the environment and the components involved in the most obvious order.

Throughout some of these tests you may resolve the problem, for example you might find if you try and push the sofa up a ladder and through the upstairs window, you can get it in the house (rather you than me), or if you try restarting the router that your Internet connection resumes. But what if the first level tests did not fix the problem or did not give any clue as to how to resolve it?

 © RM 2010

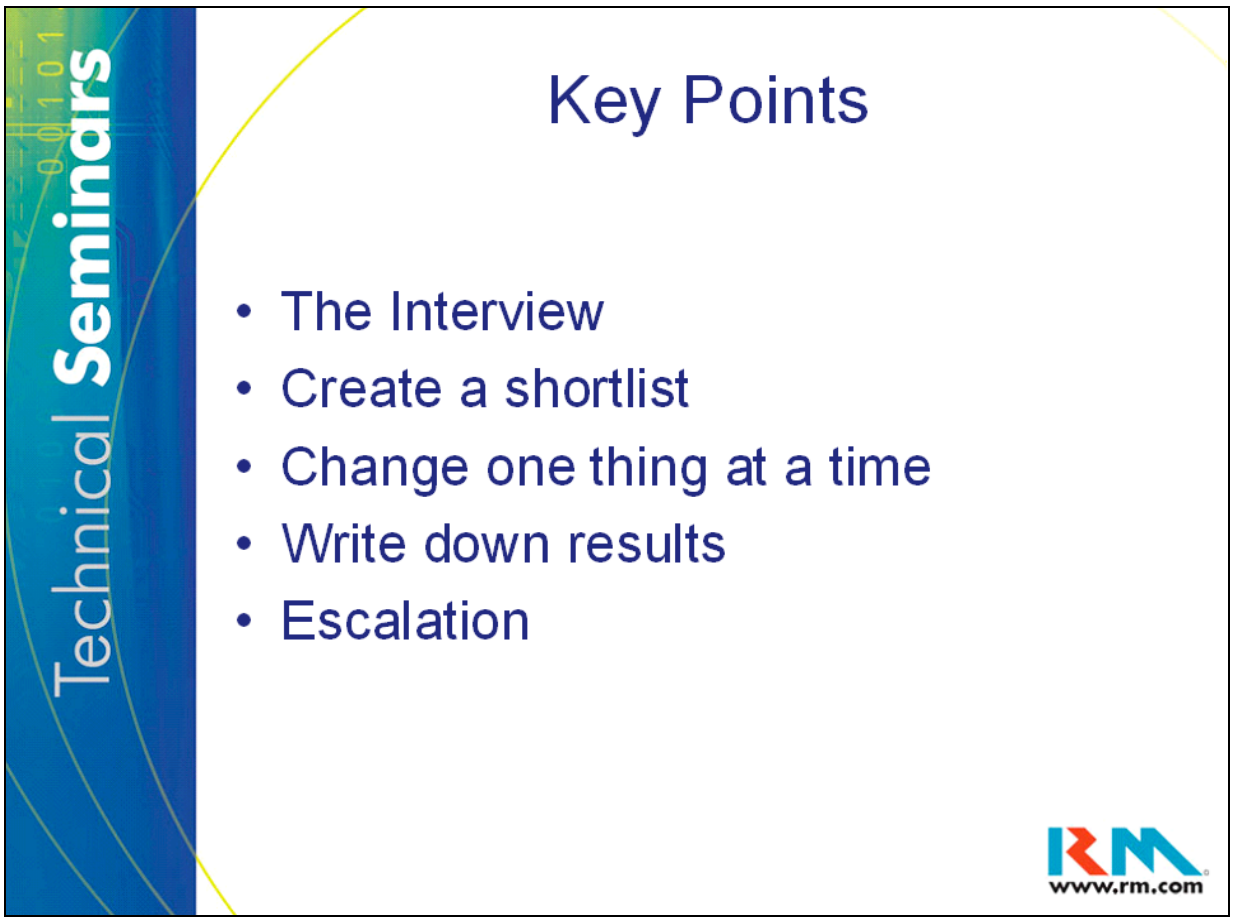
The next step is to revert back to your information review but this time thinking about the results of your testing, look to see if you missed anything. You then need to create a new list of tests or actions, thinking more creatively about solutions. I hate to use the phrase but 'thinking outside the box' is a way of putting it; after we have checked the obvious it's time to think harder about the cause.

Scenario	**Test**
TV does not display channels	Throughout the first tests you should have narrowed down if that fault was with your TV, the signal provided by the supplier or your digital box. Whichever of these always retains the fault should then be investigated in isolation. This may mean contacting the manufacturer.
Computer will not connect to the Internet	If you are still unable to connect to the Internet after checking the first level checks. You may want to contact your Internet service provider to check your line or if there are service problems. You may think about checking your computer hardware. If possible trying another computer from the same Internet/network connection would confirm if you have a computer hardware fault.
New sofa will not fit through the front door	So, it may be that your new sofa just doesn't fit through any of the entrances to your house. At this point you need to think about what you can do. Contact the supplier and advise of your situation, ask if you can exchange the sofa for a smaller one or one that breaks down into smaller pieces. After you've checked everything else that is probably your last option, always remember to measure the door before going to the sofa shop. Did I mention this happened to me?

These examples are a little simple I know, but I wanted to demonstrate how you can apply the same logic and thought process to a range of different problems.

 © RM 2010

© RM 2010



Key points to remember

To summarise this session, I was aiming to get you to think more about how you address issues and faults found while managing your network.
A lot of fault finding in the IT industry relies on experience and personal knowledge. What I have shown you in this session is a set of skills which you can apply to nearly any fault or problem. Using this technique will mean you are able to rule out suspects quicker, be able to run accurate tests which will return true or false results, document your testing results which helps with testing reviews and for future reference. All of this will help you greatly in improving your own experience and personal knowledge.

So let's have a quick look at the 5 key points I would like you to remember from this session. These are the things I would always encourage you to consider when troubleshooting any issue.

The interview

Work out what it is you are trying to find out from the conversation.
Write down your list of questions.
Don't be confrontational, you need their help!
Make sure you get all the information you need before moving on.

Create a shortlist

 © RM 2010

Rule out what definitely isn't causing the issue.
Collate a list of possible suspects.
Work out the testing order, easiest to test first.

Only change one thing at a time

Change one thing, test it, and change it back.
Be thorough.
Don't assume anything.

Write down your results

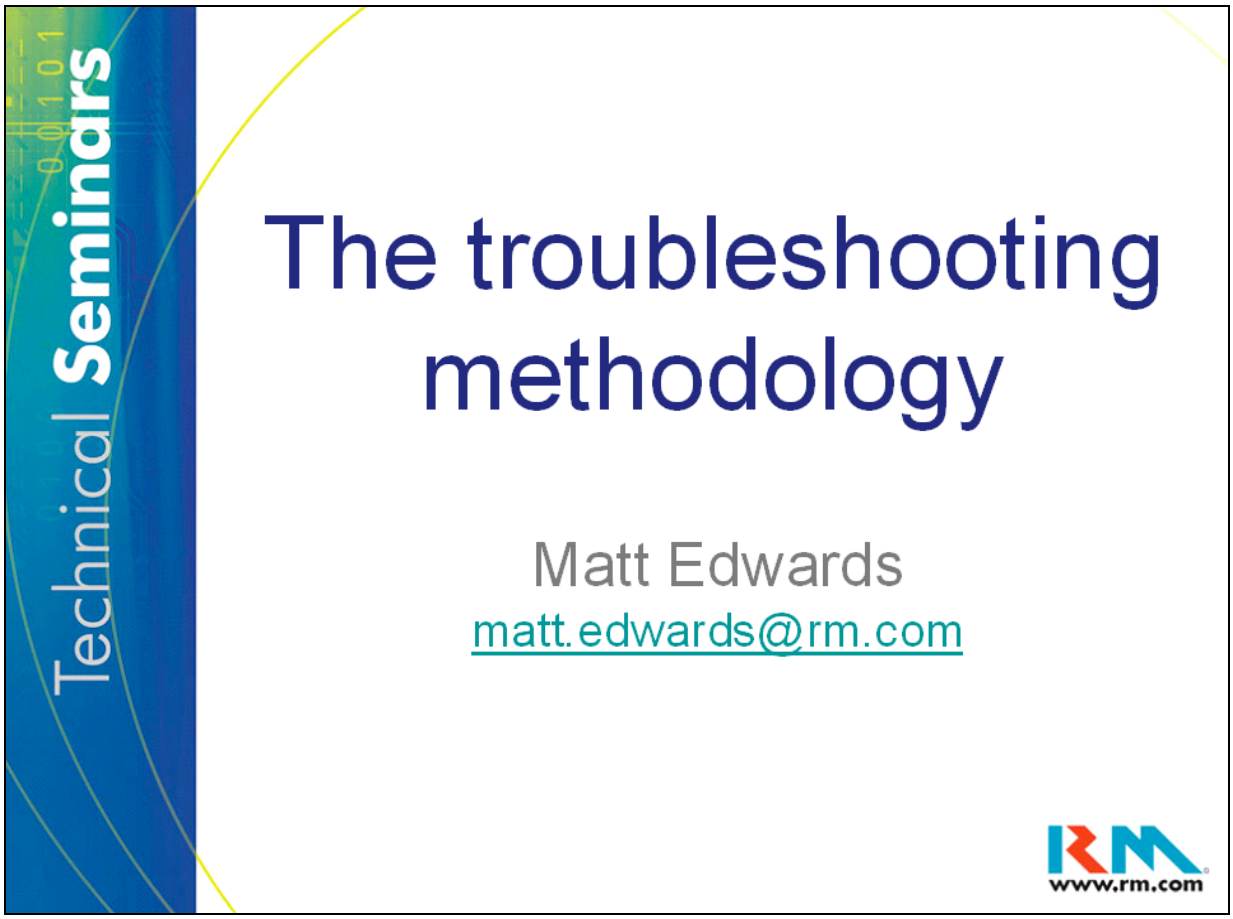
Write down testing results.
Use them to review testing and where to go next.
Once the solution is found, document the symptom, testing plan and fix. Very useful for future reference.

Escalation

Always be considering escalation, at what point do I escalate?
Be clear on the actions you need to carry out before escalating. This should come from having a clear information review and a clear test plan.
Leave your ego at the door!

So that's all, I hope this has been a useful session and has got you thinking about how you could use the troubleshooting methodology to diagnose issues within your establishments.

 © RM 2010


Technical Seminars
The troubleshooting methodology
Matt Edwards
matt.edwards@rm.com
www.rm.com

© RM 2010

© RM 2010